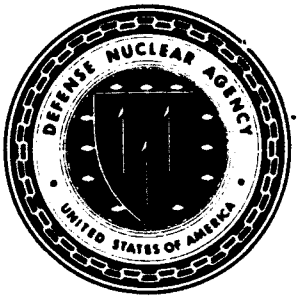
Defense Nuclear Agency
Alexandria, VA 22310-3398



DNA-TR-94-58

ICCanvas 3D Visualization Tool for Windows™
Software User's Manual

Ernie Wright
Virtual Image Labs, Inc.
1738 Elton Road
Suite 307
Silver Spring, MD 20903

October 1994

Technical Report

32 pg

CONTRACT No. DNA 001-92-C-0173

Approved for public release; distribution is unlimited.

Destroy this report when it is no longer needed. Do not return to sender.

PLEASE NOTIFY THE DEFENSE NUCLEAR AGENCY, ATTN: CSTI, 6801 TELEGRAPH ROAD, ALEXANDRIA, VA 22310-3398, IF YOUR ADDRESS IS INCORRECT, IF YOU WISH IT DELETED FROM THE DISTRIBUTION LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION.

DISTRIBUTION LIST UPDATE

This mailer is provided to enable DNA to maintain current distribution lists for reports. (We would appreciate your providing the requested information.)

☐ Add the individual listed to your distribution list.

☐ Delete the cited organization/individual.

☐ Change of address.

NOTE:
Please return the mailing label from the document so that any additions, changes, corrections or deletions can be made easily. For distribution cancellation or more information call DNA/IMAS (703) 325-1036.

NAME: _____

ORGANIZATION: _____

OLD ADDRESS	CURRENT ADDRESS
_____	_____
_____	_____
_____	_____

TELEPHONE NUMBER: ( ) _____

DNA PUBLICATION NUMBER/TITLE	CHANGES/DELETIONS/ADDITIONS, etc.) *(Attach Sheet if more Space is Required)*
_____	_____
_____	_____
_____	_____

DNA OR OTHER GOVERNMENT CONTRACT NUMBER: _____

CERTIFICATION OF NEED-TO-KNOW BY GOVERNMENT SPONSOR (if other than DNA):

SPONSORING ORGANIZATION: _____

CONTRACTING OFFICER OR REPRESENTATIVE: _____

SIGNATURE: _____

CUT HERE AND RETURN

DEFENSE NUCLEAR AGENCY
ATTN: IMAS
6801 TELEGRAPH ROAD
ALEXANDRIA, VA 22310-3398

DEFENSE NUCLEAR AGENCY
ATTN: IMAS
6801 TELEGRAPH ROAD
ALEXANDRIA, VA 22310-3398

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	941001	Technical 920930 - 940701

4. TITLE AND SUBTITLE

ICCanvas 3D Visualization Tool for Windows™ Software User's Manual

5. FUNDING NUMBERS

C - DNA 001-92-C-0173
PE - 62715H
PR - AC
TA - CE
WU - DH328130

6. AUTHOR(S)

Ernie Wright

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Virtual Image Labs, Inc.
1738 Elton Road, Suite 307
Silver Spring, MD 20903

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Nuclear Agency
6801 Telegraph Road
Alexandria, VA 22310-3398

SPWE/Byers

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DNA-TR-94-58

11. SUPPLEMENTARY NOTES

This work was sponsored by the Defense Nuclear Agency under RDT&E RMC Code Code B 4662 D AC CE 00008 7010 A AC 25904D.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The ICCanvas 3D visualization program was developed to simplify the study and communication of the results of complex meteorological models and other 3D simulations by exploiting the increasing graphics capabilities of microcomputers. The program allows you to use your personal computer to view level surfaces on structured 3D data, much of which may have been produced by large hydrocodes running on workstations and supercomputers.

ICCanvas has been used by DNA to display density distributions of dust and water vapor in 3D simulations of nuclear blast clouds. Together with sophisticated 3D rendering applications, ICCanvas has helped to produce high-quality animated views of the development of a blast cloud featuring footprint projections and multiple semitransparent surfaces.

14. SUBJECT TERMS			15. NUMBER OF PAGES
Visualization	3D Data	ICCanvas	30
Graphics	Microcomputers		16. PRICE CODE

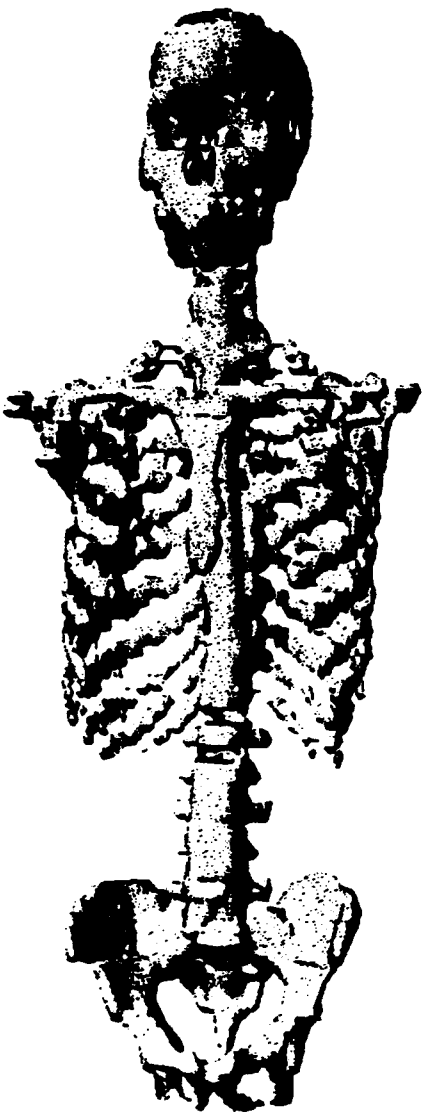
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

CLASSIFIED BY:	N/A since Unclassified
DECLASSIFY ON:	N/A since Unclassified

Table of Contents

SECTION 1

Introduction



This document describes the installation and use of the ICCanvas 3D visualization program, an interactive graphics tool that runs in the Microsoft Windows environment. ICCanvas was developed to simplify the study and communication of the results of complex meteorological models and other 3D simulations by exploiting the increasing graphics capabilities of microcomputers. The program allows you to use your personal computer to view level surfaces on structured 3D data, much of which may have been produced by large hydrocodes running on workstations and supercomputers. The ICCanvas compressed binary file format for 3D data is up to 100 times more compact than a simple text representation of the same data, meaning that formerly unwieldy 3D data sets can easily be stored and transferred using inexpensive removable media and ordinary modems.

ICCanvas has been used to display density distributions of dust and water vapor in 3D simulations of nuclear blast clouds and to reconstruct the surface geometry of an anthropomorphic phantom's skeleton from computed tomography images. Together with sophisticated 3D rendering applications, ICCanvas has helped to produce high-quality animated views of the development of a blast cloud featuring footprint projections and multiple semi-transparent surfaces. These animations were produced on videotape using exactly the same software and techniques currently being employed to create special effects for several commercial television programs.

SECTION 2

Installation

System Requirements

ICCanvas requires Microsoft Windows version 3.1 or later. Recommended: an 80386 (or better) processor, a math coprocessor, at least 4 megabytes of memory, a hard disk with at least 3 MB of free space, and a mouse.

Hard Disk Procedure

The ICCanvas program package consists of the files ICCANVAS.EXE, ICLIB.DLL, and ICHELP.HLP. To install ICCanvas, copy these three files from the distribution disk to any convenient place on your hard drive. The only restriction is that ICLIB.DLL and ICHELP.HLP must be in the current Windows search path when ICCanvas is launched. The easiest way to ensure this is to put all three files in the same directory on your hard drive, as illustrated in the following example.

Assuming your hard drive is C and the ICCanvas distribution disk is in drive A, execute these commands at the DOS prompt to copy the package to your hard drive:

```
C:\> md iccanvas                    create an ICCanvas directory
C:\> cd iccanvas                    make it the current directory
C:\iccanvas> copy a:iccanvas.exe    copy the program
C:\iccanvas> copy a:iclib.dll       copy the DLL
C:\iccanvas> copy a:ichelp.hlp      copy the help file
```

If you would like to install the data files from the distribution disk, enter these DOS commands:

```
C:\iccanvas> md data                create a data directory
C:\iccanvas> xcopy a:data\*.* data  copy the data
```

Of course, you can also perform the installation from within Windows by using File Manager to create directories and copy files. To make the ICCanvas icon visible and selectable in Program Manager, create a new program item (or group) by selecting New in the File menu of the Program Manager window. If you copied the data from the distribution disk, you will probably want to make the data directory the ICCanvas working directory.

Defining TEMP

ICCanvas creates temporary files during the construction of a surface. These files are used to collect the vertices and faces that make up a surface while the data file is being scanned. Once the surface is built, these files are no longer needed, and ICCanvas deletes them. ICCanvas asks Windows which disk should be used for temporary files. Windows

will ordinarily choose the first hard drive it finds on your system, but you may tell Windows to use another drive by defining a TEMP environment variable. (Note, however, that ICCanvas does *not* require you to do this. The procedure described in the following paragraphs may help you improve the performance of ICCanvas. If you aren't comfortable with DOS or with the techniques described, you may safely ignore this part.)

You set an environment variable by using the DOS command **set**. This command can be entered at the DOS prompt, but it usually appears as a line in your AUTOEXEC.BAT file, the batch file that is executed every time you turn on your computer. If you have set up a RAM drive, you can improve the performance of ICCanvas by making the TEMP variable point to your RAM drive. Assuming the RAM drive is D, the **set** command would be

```
set TEMP=D:\
```

If you insert this command in AUTOEXEC.BAT, Windows will tell ICCanvas to use your RAM drive for its temporary files.

There are a number of issues you should be aware of before setting a TEMP environment variable. Many Windows programs that use temporary files either look for TEMP themselves or ask Windows to do it, and in Standard mode, Windows itself may use the TEMP drive for application swap files if the WIN.INI variable **swapfile** is undefined. Your TEMP drive must be large enough to accommodate the temporary files from each of these programs, and this can be of some concern if the drive is a fixed-size RAM drive, especially since larger RAM drives reduce the amount of free memory available to Windows.

In practice, we've found that a 512K RAM drive works well and is entirely adequate for most applications. ICCanvas requires a maximum of 270K for its temporary files, and this space is only required while ICCanvas is actually building a surface. Interactions between the TEMP environment variable, RAM drives, swap files, drive caching and Windows memory management are discussed at some length in your Windows documentation.

SECTION 3

Execution Procedures

ICCanvas is an *event-driven* program. It acts only in response to requests from the user. As those with experience running Windows programs know, this makes ICCanvas more flexible than an entirely sequential program but puts greater responsibility on the user, on whom falls the task of telling the program what to do. This section therefore begins by trying to give you an idea of what the program *can* do. It also lists and describes each option ICCanvas offers you while the program is running.

Getting Started

ICCanvas allows you to visualize three-dimensional data. It assumes that the data represents a 3D array of density values that model a volume. After examining this data, it is able to build surfaces that form boundaries around higher density parts of the volume. You control the density value at which the surface is built, along with your point of view and the method used to draw the surface.

To begin, use the **Open** menu option in the File menu to open a data file. ICCanvas examines the file you choose and shows you an approximate distribution of the densities in the file using the **Level** dialog. By default, the level is set to the median value in this distribution, but you may set it to what you'd like. Densities higher than the level you choose will be inside the resulting surface.

Once the data file is chosen and the level set, ICCanvas reads the data and constructs a surface. The progress of the build is displayed, and during most of that time you have the option of canceling the build. Assuming you don't cancel, the surface is then drawn.

To view the surface from different angles, click and hold the left mouse button anywhere in the ICCanvas display. A picture of a square with a vertical line connected at one corner (the axis image) will be drawn on top of the surface. Moving the mouse will change the orientation of the axis image, and when you release the left mouse button the surface will be redrawn from the new viewing angle. For greater control over the view, use the **Camera** and **Center** menu options in the Settings menu.

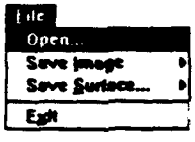
To build a surface from the same data but with a different level, use the **Level** menu option in the Settings menu. To build a surface from new data, just **Open** the new data file. Note that, unless you change it, the level setting will remain the same as it was for the old data file, and the old level may not be appropriate for the new data.

The surface can be drawn, or rendered, in one of three ways controlled by the **Render Style** menu options in the Settings menu. The ICCanvas display can be saved to a file in one of three images formats using the **Save Image** menu options in the File menu. The display can also be posted to the Windows clipboard with the **Copy Image** menu option in the Edit menu. The actual geometry of the surface can be saved in either of two formats using the **Save Surface** menu options in the File menu. Using the **Window Size** and **Colors** menu options in the Settings menu, you can precisely set the size of the display and modify the colors of the background and the "floor," a rectangle that provides a visual cue to the orientation of the surface.

The Help menu gives you access to much of the text of this section online, as well as an **About** display containing a copyright notice and acknowledgments of support for the development of the program.

Once you've finished with ICCanvas, choose **Exit** from the File menu to quit.

Open



You use the Open menu option to tell the program the name of a new IC data file. This is the first step in constructing a surface from new data. For historical reasons, IC format data files on the distribution disk all end with extensions of `.RLE`, but if you happen to have IC format files with different extensions, you can see them in the Open dialog by asking for All in the List Files of Type drop-down list. After you specify a data file and set the density level in the Level dialog, the data file will be read by the program to find and build a surface of constant density through the data.

If you choose the Cancel button in the Open dialog, the data file you were working with previously will remain the current file. If you choose OK, the current data file is forgotten and its place taken by the new one you specify. Before the program builds a surface, you'll be given a chance to change the current level. The levels listed in the Level dialog will reflect the distribution in the new file.

Each IC data file contains a 3D array of density values in a compressed binary format, along with information about the distance between data points, the origin of the data's coordinate system, the distribution of densities, the date and time when the data was created, and if appropriate, the time point the data represents. See Appendix B, IC File Format, for more information.

Save Image

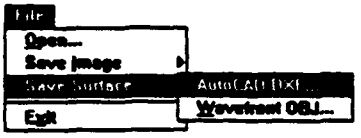
You use the Save Image menu options to create a file containing the image currently in the program's window. The image can then be loaded into other programs for editing, presentation or printing, or can be given to someone else. The size of the image and the number of colors it contains are determined by your display hardware and the Windows video driver you are using. Images can be saved in one of three formats, each of which has a broad base of support among graphics applications for personal computers.

Windows BMP. Files of this type contain Device-Independent Bitmaps, a format introduced in Windows 3.0 and based on the bitmap format used in OS/2 Presentation Manager 1.1. Some support for BMPs is built into Windows, so many Windows programs can read and write them. ICCanvas writes uncompressed BMPs.

PC Paintbrush PCX. This is a format created for ZSoft's PC Paintbrush and widely supported by both DOS and Windows programs. ICCanvas writes PCX version 5, which, unlike earlier versions, allows 8- and 24-bit images. Because PCX uses compression, this format produces files that are almost always significantly smaller than BMPs.

Macintosh PICT. The PICT format is built into the Apple Macintosh operating system. ICCanvas writes pixels in 0x009A (24-bit) and 0x0098 opcode blocks, both of which are compressed.

Save Surface



You use the Save Surface menu options to create a file containing the geometry of the current surface. ICCanvas defines a surface as a mesh of triangles, each of which is determined by the (x, y, z) positions of its three vertices. Unlike the Save Image options, Save Surface preserves all of the 3D information needed to reconstruct the surface and view it from any angle. These geometry files can be loaded into CAD programs or programs that perform photorealistic rendering, scientific visualization, or finite element analysis.

AutoCAD DXF. This is the format used by 3D applications from Autodesk and widely supported by other CAD and 3D rendering programs. Its wide support, in fact, is the primary reason for its inclusion in ICCanvas. You may find that in other respects (unwieldy file sizes for typical IC surfaces, for instance) DXF isn't entirely satisfactory.

Wavefront OBJ. OBJ is the text version of geometry files used by Wavefront Technologies' Advanced Visualizer and other programs. Support for this format will be found predominantly on Silicon Graphics workstations and in some 3D modeling and rendering programs for the Apple Macintosh, but programmers who write their own

applications to read ICCanvas geometry files will prefer the simplicity and relative speed of this format to DXF.

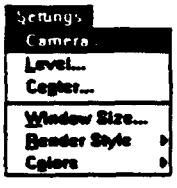
Copy Image

You use the Copy Image menu option to put a copy of the image currently in the display into the Windows clipboard. You can then switch to a different program and use that program's Paste function. This is an easy way to put ICCanvas graphics into a paint program, a presentation, or a word processing document.

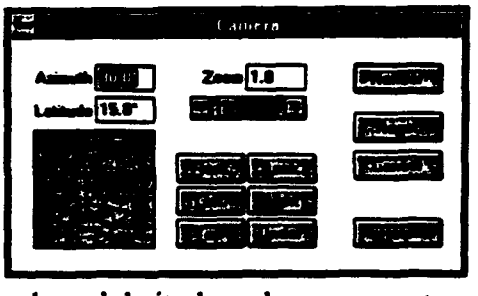
Note that this will work even if you quit ICCanvas before performing the paste. ICCanvas puts a permanent copy of the image into the clipboard; it has a life of its own and doesn't require ICCanvas for rendering.

As is customary with the clipboard, ICCanvas deletes the previous clipboard contents before posting its image. This means you should only Copy if you no longer need what is already there, including previous ICCanvas images.

Camera



You use the Camera dialog to control the position and orientation of the ICCanvas camera. The "camera" is just a convenient metaphor that disguises many of the details of the viewing system's geometry, a collection of vectors and transformations that determines what gets drawn in the display.



Azimuth, Latitude. These are used to move the camera around the current surface. A change in azimuth is equivalent to east-west motion around a globe, while a change in latitude is equivalent to north-south motion. The axis image (the picture of a square patch of ground with a vertical axis attached at the origin) will change to reflect azimuth and latitude values you enter. Conversely, moving the axis image with the mouse will change the values in the text boxes

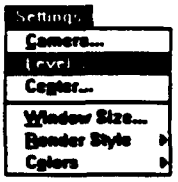
Zoom. As the name implies, this setting allows you to magnify or reduce the image in the display. Unlike a real camera, this setting has no effect on the amount of perspective in the image.

Presets. Six buttons are provided for special camera settings that align the view with an axis, set the Zoom to 1.0, and set the Center to the center of the data space. These are

useful as standard points of view, and may also be helpful after a long series of camera manipulations have left you a little uncertain about where you are.

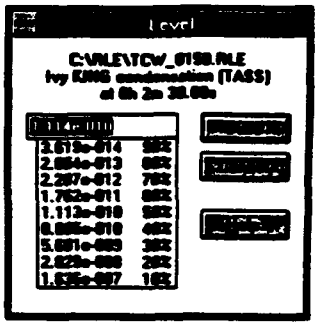
The camera dialog is available regardless of whether you have a surface to be displayed. If there is a current surface, the text of the OK button will change to Render, a reminder that accepting the changes you make will cause the display to be redrawn using the new settings. While the Camera dialog is displayed, you may press the Reset button at any time to restore the settings to what they were when the dialog was first opened.

The azimuth and latitude settings can be changed without using the Camera dialog. If you click and hold the left mouse button anywhere in the ICCanvas display, the axis image will appear. For as long as the left mouse button is held down, you may move the axis image by moving the mouse. You accept the new camera orientation when you release the left mouse button.



Level

You use the Level dialog to specify the density at which to build an isosurface. Once a level is defined, ICCanvas will scan the current data file (the one you most recently opened) to find the path of a surface of constant density at that level. The volume enclosed by this surface contains all the data points with values at or above the level you choose.

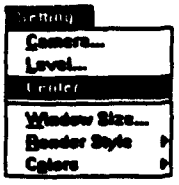


ICCanvas provides a table of levels that tries to give some indication of the distribution of densities in the data. Each level is paired with a percentage that tells how many of the *non-zero* data points will be enclosed by a surface at that level. In this respect, ICCanvas assumes that the data represents a mass of some arbitrary shape sitting in a rectangular void. The statistics in the list attempt to be relevant to the mass and not to the void.

You can select one of the listed levels or type one of your own. The value you enter must be an actual density value and not a percentage. It may be entered either in the power-of-10 notation used in the list or as an ordinary decimal number, but for most purposes it should be in the range offered by the list, since this indicates what values are in the data. Note that levels are expressed in arbitrary units that depend on the data file.

The level you set will remain in effect until the next time you set a level. You may find that a level you enter will be slightly altered the next time you see it. ICCanvas works with densities as 5-place common logarithms, and this will result in round-off errors that are harmless but may occasionally look peculiar. For a more technical explanation, see the IC File Format topic.

Center



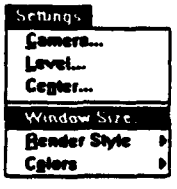
You use the Center menu option to center the display around a point you choose. When you activate centering, the mouse cursor changes to a plus image and a small window opens to show you where you are in object coordinates. To move the center of the display, move the cursor to the new center point and click the left mouse button. To cancel centering with no change, click the right mouse button.

Internally, ICCanvas treats the center as a 3D point used to position the camera. The center is the origin of the azimuth and latitude rotations, or the pivot point around which the camera moves. You can get complete control of the 3D position of the center point by using Center in combination with the preset orientations in the Camera dialog, in the following way:

1. In the Camera dialog, choose the Top or Under view to get an x-y projection.
2. From the Settings menu, choose Center to set the x and y coordinates.
3. In the Camera dialog, change the latitude to 0° to get a projection that includes z. Leave the azimuth at 0°.
4. Choose Center again to set the z coordinate.

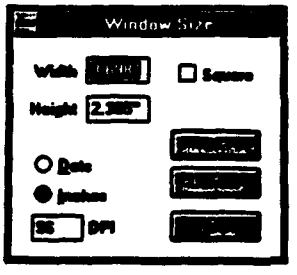
You should avoid using the preset orientations once you have a center that you want to keep. The presets automatically restore the center point to the center of object space, which is why they weren't used in step (3) above.

Window Size



The Window Size dialog allows you to precisely set the size of the ICCanvas display. This is useful for controlling the size of the images created by the Save Image and Copy menu options. Window Size controls the dimensions of the display area (in Windows parlance, the *client area* of the window), and not the entire window.

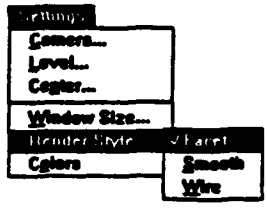
You can set the size in either pixels (picture dots) or inches. Sizes expressed in inches depend on a scale factor, the number of pixels per inch. The video driver for your display provides a default scale that is displayed in the DPI (dots per inch) text box the first time the Window Size dialog is opened. Changing the



DPI setting affects the scaling ICCanvas performs to determine pixel dimensions from sizes expressed in inches.

Suppose you would like to produce a 3" × 3" ICCanvas image for inclusion in a word processing document. Your word processor allows you to set the DPI for an image, and you know you can get good results from your printer if images are printed at 150 DPI. In ICCanvas, you can set the DPI to 150 and the window dimensions to 3" × 3", and the display will be sized to the correct pixel dimensions.

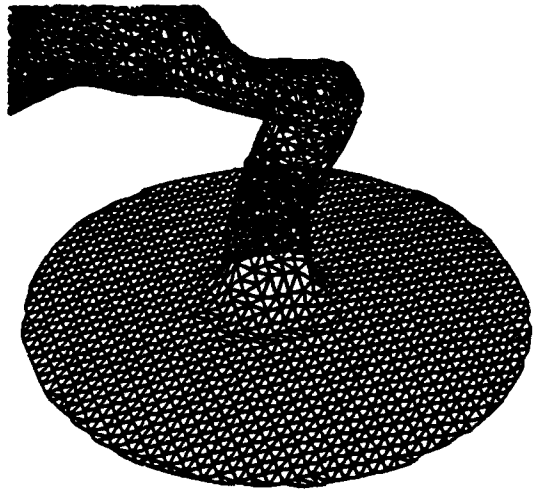
Note that the DPI setting is internal to ICCanvas. It has no effect on Windows, nor is it explicitly written into image files you save or images you copy to the clipboard.



Render Style

You use the Render Style menu option to set the way the surface is drawn in the ICCanvas display. The choice involves a tradeoff between the quality of the image, the information it contains, and the time it takes to draw. The new render style will be employed the next time ICCanvas draws the surface. To see the change immediately, click the left mouse button anywhere in the ICCanvas display.

Facet. This option uses constant shading across each of the triangles that make up the surface. The rendering is relatively fast, but the simple shading emphasizes the polygonal nature of the surface approximation, which may distract you from features of the surface that should appear continuous. On displays with 16 or fewer colors, facet rendering uses the ordered dither method of color approximation built into Windows, and this may also be distracting.

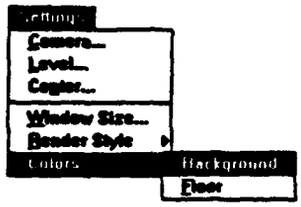


Smooth. This option employs Gouraud interpolated shading to draw the surface triangles. Interpolated shading ensures that the shading of the surface is continuous across triangle edges. If the polygon mesh is an approximation of a smooth surface, continuous shading across polygon boundaries provides a better representation of the underlying surface. Gouraud shading works at the level of picture dots, or pixels, rather than triangles, so it can be somewhat slower. In ICCanvas, it also uses absolute color, and on displays with 16 or fewer colors this can mean that the surface is drawn with only the three or four shades of gray that happen to be available.

Wire. This option draws a wireframe version of the surface. Wireframes make the polygon representation of the surface explicit and may be helpful in clarifying the shape of certain features in cases where the shaded renderings appear ambiguous. This is also the fastest rendering method.

The shading of the surface for the Facet and Smooth options uses the Lambert relation for diffuse reflected light, in which the apparent brightness of a face is proportional to the cosine of the angle formed by the surface normal and vector pointing toward the light source. In ICCanvas, the light source is always coincident with the camera.

Colors



You use the Colors menu options to set the colors of the background and the floor in the ICCanvas display. This is useful if you plan to print ICCanvas images in black and white and would like a white background, for instance. You may also find that the default ICCanvas colors are not good choices for your particular display, or that they are just unappealing to you.

Color changes remain in effect only for the current ICCanvas session.

SECTION 4

Error Messages

This section lists the error messages you might see while using ICCanvas. The messages are listed in execution order—messages for errors that would occur early in an ICCanvas session are listed first.

- **File Error: Cannot find ICLIB.DLL**
- **File Error: Cannot find COMMDLG.DLL**

When Windows starts ICCanvas, it also tries to open ICLIB.DLL and COMMDLG.DLL, since ICCanvas requires both files in order to run. ICLIB.DLL is the IC dynamic-link library and contains much of the functionality of the program. COMMDLG.DLL is part of the Windows 3.1 package and contains the Open and Choose Color common dialogs Both files must be in the Windows or Windows\System directories, or in the same directory as ICCANVAS.EXE, when you start ICCanvas. See the Installation section for more information.

- *Filename*
  Cannot find this file.
  Please verify that the correct path and filename are given.

The filename you entered in the Open dialog can't be matched to an existing file. If the problem with the name of the data file isn't obvious, try entering the name by picking the path and file from the lists provided in the Open dialog, rather than typing it into the filename edit field.

- **Unable to open "*filename*".**

Windows thinks that the data file you chose from the Open dialog exists, but ICCanvas is unable to open it. This won't happen often, although it's possible that read access to a file might be denied because, for example, another application has the file open.

- *Filename*
  is not an IC format file.

The file you chose from the Open dialog is not a legal data file. ICCanvas checks the file you name to ensure that it is large enough to be a data file, that the name field is a legal C-language string, and that the dimensions are reasonable. This will prevent you, in most cases, from accidentally loading a non-IC file as a data file.

- **General Protection Fault.**

Windows displays this message when an application has attempted to read from or write to memory it doesn't own. In ICCanvas, this is usually a symptom of a bad data file (a damaged or non-IC file), especially if it occurs during the construction of a surface. In any case, this is a serious message, and without knowing the cause, you shouldn't assume that any part of Windows will function reliably once you have received it—while protected

mode prevents most kinds of illegal addressing, this is only protection against a symptom and not whatever pathology caused it. It is safest to exit and restart Windows.

- **The surface is too large at the specified Level.**
  Try using a higher (less inclusive) value.

Surfaces cannot contain more than 32,767 triangles and 16,384 points, a practical limit that arises both because of the way PCs use memory and as a judgment about what ICCanvas can reasonably do. This message may also result from the use of an inappropriate level left over from a previous surface. Guided by the density distribution displayed in the Level dialog, change the level to one that will create a surface with fewer components and try the build again. Note that, until the build is successful, ICCanvas has no surface to display.

- **No surface exists at the specified Level.**
  Try using a lower (more inclusive) value.

The build produced no triangles, implying that all of the data values are less than the current level. This message may result from the use of an inappropriate level left over from a previous surface. Guided by the density distribution displayed in the Level dialog, change the level to one that will find a surface and try the build again. Note that, until a non-empty surface is built, ICCanvas has no surface to display.

- **ICLIB: icBuildInit() Opening *or* Reading source.**
- **ICLIB: icBuildInit() Opening temp files.**
- **ICLIB: icBuildInit() Memory allocation.**
- **ICLIB: icBuildInit() Writing vertices.**
- **ICLIB: icBuild() Reading source *or* Writing vertices *or* faces.**
- **ICLIB: icCompactFaces() Memory allocation or file i/o.**
- **ICLIB: icBuildFinish() Memory allocation.**
- **ICLIB: icBuildFinish() Reading vertices *or* faces.**

These errors can occur during the construction of a surface, and all indicate a problem with a disk or memory resource. A memory allocation error usually means that you have run out of free memory, but it may also be caused by memory *fragmentation*—Windows tries very hard to rearrange items in memory so that the remaining free memory is in a small number of large, contiguous blocks, and for various reasons it may not always succeed. ICCanvas can require as much as 1.5 MB (megabytes) of memory, in separate pieces that are never larger than 64K (kilobytes) each. If you are running other programs concurrently with ICCanvas, try quitting from those programs and restarting the build. As a last resort, you can check for memory fragmentation by quitting and restarting Windows itself, but be aware that a program in your Windows startup—a screen saver, for example—might be the source of the problem.

Disk errors are less common and can be caused by a loss of access to a disk or file, by a full disk, or by a hardware problem. Loss of access can occur when files are being read from or written to removable media (floppy disks, hard disk cartridges or tape, for

instance) or storage accessed over a network. ICCanvas may also be attempting to create its vertex and face temporary files on a write-protected or nearly full disk. During the build, ICCanvas uses up to 270K of disk space for its temporary files. These files are deleted when the build is complete.

The disk used for temporary files is determined by Windows, but you may tell Windows to use a particular drive for temporary files by defining a TEMP environment variable. See the Installation section. Note to programmers: the IC DLL relies on the API function GetTempFileName() and does not attempt to validate the name or check disk free space.

- **Unable to save the image.**
- **Unable to save the surface.**

These messages can appear during processing of the Save Image and Save Surface menu options. The errors that cause them are of the same type as those that may occur during surface construction: a lack of memory or a problem with the output file. In particular, make sure that the filename you supply for the save operation is valid—the disk exists, it's inserted, and it isn't write-protected, the path exists and is typed correctly, the base name isn't more than 8 characters long, the extension isn't more than 3 characters long, all the characters are legal for filenames, you aren't trying to overwrite an existing file with its write/delete protection bit set, etc. Windows should alert you to some of these problems before the program has gotten far enough to issue these errors, but make no assumptions.

Also make sure that the disk has enough free space for the file. Surface files, in particular, can be quite large, requiring more space than is available on an empty high-density floppy in some cases.

APPENDIX A

The IC Approach

This appendix gives a brief development of level surfaces and discusses the method IC uses to build isosurfaces.

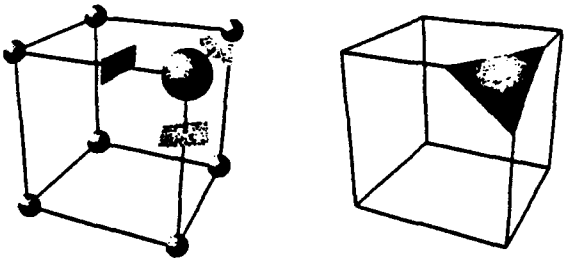
ICCanvas and the IC DLL construct what are called *level surfaces* or *isosurfaces*. For any number c, the set of points (x,y,z) for which $f(x,y,z)=c$ is called a level surface of f. If $f(x,y,z)$ gives the temperature at points (x,y,z) in space, the level surface $f(x,y,z)=c$ is the set of all points at which the temperature is c and is called an isothermal surface. Similarly, $V(x,y,z)=c$ might be a surface of constant voltage, called an equipotential surface.

Spheres centered at the origin are a set of level surfaces of the function

$$f(x,y,z)=x^2+y^2+z^2$$

since $x^2+y^2+z^2=c$ is an equation of such a sphere for non-zero c. In this example, the isosurface can be constructed analytically because the function is known. IC, on the other hand, works with function values tabulated at regular intervals (i_x,i_y,i_z) and arranged in a 3D array, or scalar field. See the IC File Format appendix for information on the way scalar fields are presented to IC as data files.

IC extracts surfaces by scanning a scalar field two slices at a time, from the bottom up. Each adjacent pair of slices forms a lattice, with data values where the grid lines meet (or, the other way around, each data point not on a slice edge is connected by grid lines to four of its neighbors on its own slice, plus the point above or below it). The lattice divides the space of the scalar field into cubes, each of which has 12 grid line edges and 8 data point corners.



For each cube, IC determines whether each of its eight corners is inside or outside the level surface. This information is used to infer the path of the surface. In particular, if some corners are inside and some outside the surface for a given cube, the surface is assumed to pass through the cube. Every grid line shared by two data points on opposite sides of the level contributes a point on the surface where it cuts the grid line.

An 8-bit binary number is sufficient to describe every possible arrangement of inside and outside corners and therefore every possible inference about the path of the surface through a cube. IC uses such a number as an index into a table of triangular surface patches. The triangle vertices of the patches all lie on grid lines with inside/outside corner

pairs, at points where the surface cuts the grid lines. The exact location of the vertices is determined by linear interpolation between data points and along grid lines.

For each combination of scalar field and level, IC ends up with a list of triangles that approximate the level surface. ICCanvas then uses the 3D graphics functions in the IC DLL to present interactive views of this surface to the user. Consistent with IC's design goals, the ICCanvas display is fast and simple. Both the display and the surface geometry can be saved to files and loaded into other programs that provide more sophisticated visualization tools, including animation, photorealistic rendering and multiple semitransparent surfaces.

APPENDIX B

IC File Format

This appendix discusses the binary format for input files used by ICCanvas and the IC DLL. It assumes a knowledge of programming and some experience with reading and writing binary files. A knowledge of the C language and of 3D graphics concepts will be helpful.

The binary format of simple data types in an IC data file is based on the internal formats of most popular C compilers running on Intel 80x86 PCs. The byte order places the most significant byte at the highest linear address, an order sometimes called big-endian. As used here, the C types `short` and `long` are 2- and 4-byte two's complement integers. The C type `float` is a 4-byte IEEE floating-point number. The header structure described below is packed, meaning that no extra bytes are allowed between structure members. The file format, however, aligns all data in the file on 2-byte boundaries.

IC data files consist of 3D density data preceded by a 118-byte header. The data values are assumed to be arranged in a 3D lattice or structured grid (picture the steel skeleton of a building). Each data value is associated with a point in space where the grid lines cross. Data points are at equal intervals for a given axis, but for different axes the spacing can be different. The header, defined below, gives the size of the grid, the spacing of the data points along each axis, and other information about the data.

```
typedef struct {
   unsigned short    size;
   float             scale;
   float             offset;
} AXIST;

typedef struct {
   char              comment[ 40 ];
   char              reserved;
   unsigned short    percentile[ 9 ];
   long              creation_date;
   float             simulation_time;
   AXIST             axis[ 3 ];
} HEADER;
```

comment — A 40-character NULL-terminated string. Intended for use as a name or description field, this string is displayed in the Level and Progress dialogs. There is no restriction on its format.

reserved — This space is currently ignored; you may hide what you want here.

percentile — An array containing an approximate distribution for the densities. At each decile interval from 90% to 10%, the array gives the level at which that percentage of the non-zero data points will be enclosed in an isosurface at that level. This is the information displayed by the Level dialog. Levels are floating-point values

written in the same 16-bit format described below for the density data.

creation_date This is a creation date and time written as a 4-byte integer in the manner of the ANSI function time(), which returns the number of seconds since 00:00:00 January 1, 1970. It is meant to document when the original data was created, but may also be used to record when the IC format file was generated from the data.

simulation_time If appropriate for the data, this stores the time in floating-point seconds from the time origin. For simulations that produce data at a sequence of time points, this field can be used to give the time coordinate. The simulation time is displayed in the Level and Progress dialogs.

axis An array of three structures giving the dimensions (**size**), distance between data points (**scale**), and **offsets** from the origin. The scale and offset values are in arbitrary units. Having a scale for each axis allows ICCanvas to anisotropically scale its surfaces for cases where the volumetric cells, or voxels, from which the data have presumably been sampled are not cubes. The offsets allow ICCanvas to preserve spatial information in cases where a simulation has tracked a moving mass to keep it within the bounding box of its space. Both scale and offset are taken into account when a surface is saved with the Save Surface menu options.

The header is followed by the actual density data. Densities are expressed in arbitrary units, although by convention they have been in grams per cubic centimeter. Each density value is a floating-point number packed into 16 bits by the following expression:

```
x <= 1e-16 ? 0 : 2048 * log10( x ) + 32768
```

Packed densities may range from 1e-16 to 1e+16, with a precision of about 5 decimal digits in log space. The special value 0 is reserved for "empty" parts of the volume. The advantages of this encoding are that we preserve relative precision over a large range, we store only the precision we can use, and we are able to treat the density values internally as integers, improving performance and conserving memory.

Once packed, the densities are run-length encoded. RLE eliminates redundancy in the data array by replacing runs of three or more equal values with a count followed by a single instance of the value. Each *x-y* plane, or slice, of densities is encoded separately (runs don't cross slices) and is preceded in the file by a two-byte size field containing the size of the slice, in short integers, after encoding.

BASIC and C source code for producing IC format data files is included on the distribution disk. The C version includes several machine-independent functions for

writing the binary images of simple data types, as well as a function called Pack() for performing run-length encoding. A simple implementation that doesn't require RLE can be written on the basis of the information given here. When writing densities, write for each slice the 2-byte size `ISIZE*JSIZE+1`, followed by a single 2-byte RLE code `ISIZE*JSIZE-1`, followed by the density values for the slice. The RLE code tells ICCanvas to treat the entire slice as one verbatim block.

APPENDIX C

ICMAKE.BAS

This appendix lists a BASIC program for producing data files in IC format. To run the program, type QBASIC at the DOS prompt. This starts the BASIC interpreter included with versions 5.0 and later of MS-DOS. Open ICMAKE.BAS, the text of the program, from the SOURCE\ICMAKE directory on the distribution disk, then press F5 (Run). A sample input file, called TEST.DAT, is included on the distribution disk.

```
' ICMAKE.BAS ------------------------------------------------------
' EW  22 Mar 94
'
' A BASIC program for creating data files in IC format.
'
' This program demonstrates the simplest approach to the creation of
' binary IC input files using the Microsoft QBASIC interpreter, part
' of MS-DOS 5.0 and later.  ICMAKE.BAS reads a text file containing
' 3D density data, preceded by header items describing the data:
'
'    name or comment text, less than 40 characters, on line 1
'    a simulation time in seconds, or 0.0, on line 2
'    for each axis:  an array dimension, distance between data points,
'       and origin coordinate on lines 3, 4 and 5
'    density values separated by commas
'
' It then writes an IC binary file.  Because of the relative slowness
' of interpreted BASIC, run-length encoding is not done.  Each slice
' is instead preceded by a single RLE code that tells IC that the
' entire slice is in the file verbatim.
'
' This program may be tested by using the example text file called
' TEST.DAT as the input file.
' ------------------------------------------------------------------


' ---------- IC header definition
TYPE AxisT
   size        AS INTEGER
   scale       AS SINGLE
   offset      AS SINGLE
END TYPE

TYPE Header
   comment     AS STRING * 40
   reserved    AS STRING * 22
   percentile AS STRING * 18       ' unsigned short[ 9 ]
   creatdate   AS LONG
   simtime     AS SINGLE
   xaxis       AS AxisT
   yaxis       AS AxisT
   zaxis       AS AxisT
END TYPE
```

```
' ---------- procedure declarations
DECLARE SUB GetHeader (hdr AS Header)
DECLARE SUB GetSlice (nf AS INTEGER, buf() AS INTEGER, size AS INTEGER)
DECLARE SUB PutHeader (nf AS INTEGER, hdr AS Header)
DECLARE SUB PutSlice (nf AS INTEGER, buf() AS INTEGER, size AS INTEGER)
DECLARE SUB MakeDist (hdr AS Header)
DECLARE SUB SumDist (hdr AS Header, buf() AS INTEGER, sz AS INTEGER)
DECLARE SUB ShowProgress (i AS INTEGER, imax AS INTEGER)
DECLARE FUNCTION LevelI2D! (i AS LONG)
DECLARE FUNCTION LevelD2I% (d AS SINGLE)


DIM k AS INTEGER, sz AS INTEGER      ' slice index and size
DIM hdr AS Header                    ' IC header
DIM perc(1024) AS INTEGER            ' density distribution
DIM nperc AS LONG                    ' number of non-zero data points


' ---------- program begins here
GetHeader hdr
sz = hdr.xaxis.size * hdr.yaxis.size
DIM src(sz) AS INTEGER
PutHeader 2, hdr

FOR k = 1 TO hdr.zaxis.size
   GetSlice 1, src(), sz
   SumDist hdr, src(), sz
   PutSlice 2, src(), sz
   ShowProgress k, hdr.zaxis.size
NEXT k

MakeDist hdr
PutHeader 2, hdr

END
' ---------- of ICMAKE


SUB GetHeader (hdr AS Header)
   DIM s AS STRING

   PRINT "--------------------"
   PRINT "       IC Make"
   PRINT "--------------------"
   INPUT "Input File......", s
   OPEN s FOR INPUT AS 1
   INPUT "Output File.....", s
   OPEN s FOR BINARY ACCESS WRITE AS 2

   INPUT #1, s
   hdr.comment = LEFT$(s + CHR$(0), 40)
   INPUT #1, hdr.simtime
   INPUT #1, hdr.xaxis.size, hdr.xaxis.scale, hdr.xaxis.offset
   INPUT #1, hdr.yaxis.size, hdr.yaxis.scale, hdr.yaxis.offset
   INPUT #1, hdr.zaxis.size, hdr.zaxis.scale, hdr.zaxis.offset
END SUB


SUB GetSlice (nf AS INTEGER, buf() AS INTEGER, size AS INTEGER)
   DIM i AS INTEGER
```

```
   DIM x AS SINGLE

   FOR i = 1 TO size
      INPUT #nf, x
      buf(i) = LevelD2I(x)
   NEXT i
END SUB


FUNCTION LevelD2I% (d AS SINGLE)
   DIM i AS LONG

   IF d < 1E-16 THEN
      i = 0
   ELSE
      i = INT(2048# * (LOG(d) / 2.30258512497#) + 32768#)
   END IF

   LevelD2I = (i AND &HFFFF)
END FUNCTION


SUB MakeDist (hdr AS Header)
   DIM n AS LONG, s AS LONG, d AS LONG, j AS LONG
   DIM i AS INTEGER, j2 AS INTEGER
   SHARED perc() AS INTEGER
   SHARED nperc AS LONG

   d = nperc \ 10
   s = 0
   i = 8

   FOR n = 2 TO 1024
      s = s + perc(n)
      DO WHILE s >= d
         j = 64 * n - (64 * (s - d)) / perc(n)
         j2 = j AND &HFFFF
         MID$(hdr.percentile, i * 2 + 1, 2) = MKI$(j2)
         s = s - d
         i = i - 1
         IF i < 0 THEN EXIT DO
      LOOP
      IF i < 0 THEN EXIT FOR
   NEXT n
END SUB


SUB PutHeader (nf AS INTEGER, hdr AS Header)
   PUT #nf, 1, hdr
END SUB


SUB PutSlice (nf AS INTEGER, buf() AS INTEGER, size AS INTEGER)
   DIM rlecode AS INTEGER, i AS INTEGER

   rlecode = size - 1
   i = size + 1
   PUT #nf, , i
   PUT #nf, , rlecode
   FOR i = 1 TO size
```

```
      PUT #nf, , buf(i)
   NEXT i
END SUB


SUB ShowProgress (i AS INTEGER, imax AS INTEGER)
   CONST FGGAUGE = 178
   CONST BKGAUGE = 176

   IF i = 1 THEN PRINT STRING$(21, BKGAUGE);
   LOCATE , 1: PRINT STRING$((21 * i) \ imax, FGGAUGE);
   IF (i = imax) THEN PRINT : PRINT "Done."
END SUB


SUB SumDist (hdr AS Header, buf() AS INTEGER, sz AS INTEGER)
   DIM i AS INTEGER, j AS INTEGER
   SHARED perc() AS INTEGER
   SHARED nperc AS LONG

   FOR i = 1 TO sz
      IF buf(i) <> 0 THEN
         j = (buf(i) AND &HFFFF) \ 64
         perc(j) = perc(j) + 1
         nperc = nperc + 1
      END IF
   NEXT i
END SUB
```

DISTRIBUTION LIST

DNA-TR-94-58

DEPARTMENT OF DEFENSE

ASSISTANT TO THE SECRETARY OF DEFENSE
ATTN: EXECUTIVE ASSISTANT

DEFENSE INTELLIGENCE AGENCY
ATTN: DIW-4
ATTN: DT-2A G WEBER

DEFENSE NUCLEAR AGENCY
2 CY ATTN: IMTS
ATTN: NASF
ATTN: OPNA
ATTN: OPNA P OSWALD
5 CY ATTN: RAEM
ATTN: RAEM ROBERT KEHLET
2 CY ATTN: SPWE
ATTN: SPWE LTC MARK BYERS
ATTN: SPWE LT COL HIM HODGE
ATTN: SPWE MAJ ROB COX
ATTN: SPWE K PETERSEN

DEFENSE TECHNICAL INFORMATION CENTER
2 CY ATTN: DTIC/OC

DEPARTMENT OF THE ARMY

ARMY RESEARCH LABORATORIES
ATTN: SLCSM-SE

U S ARMY NUCLEAR & CHEMICAL AGENCY
ATTN: MONA-NU DR D BASH

DEPARTMENT OF THE NAVY

NAVAL RESEARCH LABORATORY
ATTN: CODE 7920

NAVAL SURFACE WARFARE CENTER
ATTN: CODE K42 L VALGE

STRATEGIC SYSTEMS PROGRAM
ATTN: SP0272 R G STANTON

STRATEGY AND POLICY DIVISION
ATTN: NUC AFFAIRS & INT'L NEGOT BR

DEPARTMENT OF THE AIR FORCE

AFIT/DEEE
ATTN: DR K MATHEWS

AIR FORCE STUDIES AND ANALYSIS
ATTN: AFSAA/SAS

AIR UNIVERSITY LIBRARY
ATTN: AUL-LSE

HQ USAF/XOFN
ATTN: XOFN

USSSTRATCOM
ATTN: LT COL A T HOPKINS
ATTN: MAJ T GRIFFITH
ATTN: LTC T MCGRANN

DEPARTMENT OF ENERGY

LAWRENCE LIVERMORE NATIONAL LABORATORY
ATTN: ALLEN KUHL

OAK RIDGE NATIONAL LABORATORY
ATTN: B WORLEY

OTHER GOVERNMENT

CENTRAL INTELLIGENCE AGENCY
ATTN: OSWR/NED

DEPARTMENT OF DEFENSE CONTRACTORS

ARS
ATTN: S E HOLMAN

HORIZONS TECHNOLOGY, INC
ATTN: E TAGGART

JAYCOR
ATTN: CYRUS P KNOWLES

KAMAN SCIENCES CORP
ATTN: J HESS
ATTN: R HARDY

KAMAN SCIENCES CORP
ATTN: DASIAC

KAMAN SCIENCES CORPORATION
ATTN: DASIAC

LOGICON R & D ASSOCIATES
ATTN: LIBRARY

LOGICON R & D ASSOCIATES
ATTN: J WEBSTER
ATTN: T MAZZOLA

LOGICON R & D ASSOCIATES
ATTN: G GANONG

MAXWELL LABORATORIES INC
ATTN: J BARTHEL

NEW MEXICO ENGINEERING RESEARCH INSTITUTE
ATTN: LIBRARY

PACIFIC-SIERRA RESEARCH CORP
ATTN: R SMALL

S-CUBED
ATTN: C NEEDHAM

SCIENCE APPLICATIONS INTL CORP
ATTN: D BACON
ATTN: J COCKAYNE
ATTN: P VERSTEEGEN

SCIENCE APPLICATIONS INTL CORP
ATTN: J SONTOWSKI

UNITECH
ATTN: DAVE STRAW

VIRTUAL IMAGE LABS INC
2 CY ATTN: ERNIE WRIGHT